Remarks/Arguments

Claims 1-15 remain pending in the present application. No claims were amended, no claims were added, and no claims were canceled. Applicants have carefully considered the cited art and the Examiner's comments, but continue to believe the claims currently in the case patentably distinguish over the cited art and are allowable in their present form. Reconsideration of the rejection is, accordingly, respectfully requested in view of the following comments.

I. 35 U.S.C. § 102, Anticipation

The Examiner has rejected claims 1-15 under 35 U.S.C. § 102(e) as being anticipated by Jeffords et al. (U.S. Patent No. 6,233,623). This rejection is respectfully traversed.

In rejecting the claims, the Examiner continues to state as follows:

> As to claim 1, Jeffords teaches a client server system using distributed objects, comprising: a client connected to a communication network for performing an access request to an object (col. 14, line 36-col. 15, line 17); an application server for performing an application by an actual object according to the access request by said client (col. 14, line 36-col. 15, line 17); and an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is part of said actual object (col. 14, line 36-col. 15, line 17) wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server (col. 14, line 36-col. 15, line 17).

Final Office Action dated November 3, 2004, pages 2 and 3.

Claim 1 of the present application reads as follows:

> 1. A client server system using distributed objects, comprising:
> a client connected to a communication network for performing an access request to an object;

an application server for performing an application by an actual object according to the access request by said client; and

an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server.

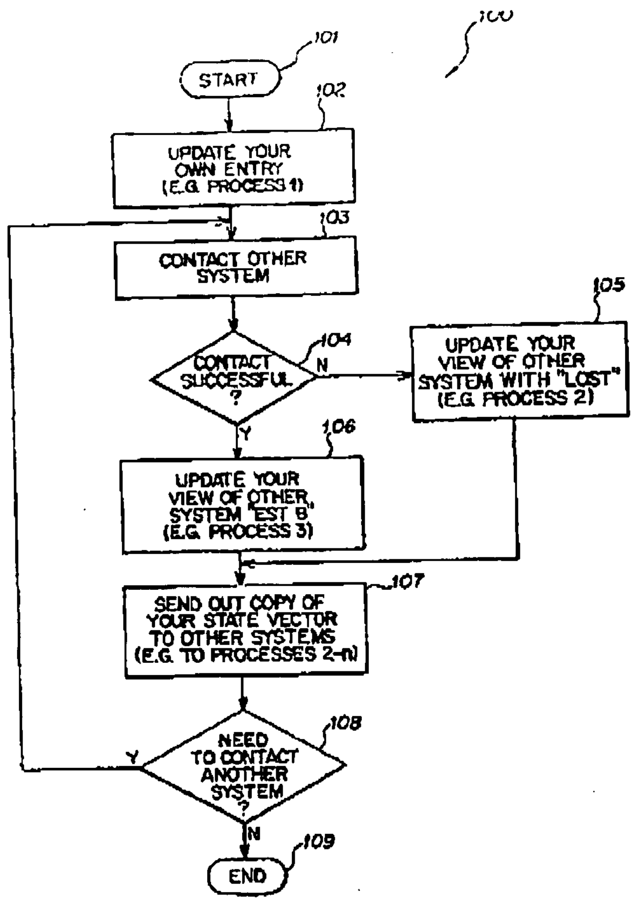
Jeffords et al. (hereinafter Jeffords) does not disclose a client server system using distributed objects that includes "an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server".

The Examiner refers to col. 14, line 36-col. 15, line 17 of Jeffords, as generally disclosing the subject matter of claim 1. Applicants respectfully disagree. Col. 14, line 36-col. 15, line 17 of Jeffords is as follows:

> 5. Updating State Information
>
> Fig. **8** is a flowchart of a method for updating state information **100**. Starting at Step **101**, a particular system, (e.g., process 1) updates its own contact status vector (Step **102**). Next, process 1 contacts the other systems (Step **103**). Further action depends upon whether the contact is successful (Step **104**); if not successful, process 1 updates its state vector with "lost" for the noncontacted system; e.g., process 2 (Step **105**). If contact was successful, process 1 updates its state vector with ESTB for the contacted system, e.g., process 3 (Step **106**). In either case, process 1 then sends out a copy of its state vector to the other systems, e.g., processes 2-n (Step **107**). Then process 1 continues to contact other systems (Step **108**); when all processes have been attempted to be contacted, the method ends (Step **109**).
>
> The RRM has many areas of potential application. For example, in a switched communications network having a distributed call management system, a resource manager can be provided for each of a plurality of distributed call

management processes. See, for example, the switched network described in copending and commonly owned U.S. Serial No. 08/188,238 filed Jan. 28, 1994 by K. Dobbins et al., hereby incorporated by reference in its entirety. A distributed memory space can be divided into separate pools for each of the following services: topology, directory, policy and call. A first benefit of incorporating RRM in this system is that every process is provided with a consistent view of the network. For example, every process can have access to a policy pool which may specify, for example, that node-a is permitted to communicate with node-b. As a further example, when a node in the network topology changes, it is important that every switch be notified of the change. Using RRM, state changes are seen every where so that every switch has the same view of the network topology. A second benefit is fault tolerance; for example, because the call pool has an object for every call, if one call management process goes down, another call management process can take over the calls in the call pool previously owned by the failed process; this can be done quickly with essentially no interruption in service. A third benefit is load sharing. The various call management processes can distribute ownership of calls in order to balance the workload. A fourth benefit is scalability; additional call processes can be added to the system easily without requiring reprogramming of the prior systems. Yet another important benefit from a programming perspective is that the programmer does not need to know where any given method or change of object is executed, only that the result is the same.

Figure 8 of Jeffords is reproduced below for the convenience of the Examiner:


100
101
START
102
UPDATE YOUR OWN ENTRY (E.G. PROCESS 1)
103
CONTACT OTHER SYSTEM
104
CONTACT SUCCESSFUL ?
N
Y
105
UPDATE YOUR VIEW OF OTHER SYSTEM WITH "LOST" (E.G. PROCESS 2)
106
UPDATE YOUR VIEW OF OTHER SYSTEM "EST B" (E.G. PROCESS 3)
107
SEND OUT COPY OF YOUR STATE VECTOR TO OTHER SYSTEMS (E.G. TO PROCESSES 2-n)
108
NEED TO CONTACT ANOTHER SYSTEM ?
Y
N
109
END

Jeffords describes a procedure in which a particular system updates its own contact status vector and then contacts other systems (steps **102** and **103** in **Figure 8** of Jeffords). If contact is made with another system, the particular system updates its state vector with ESTB (contact with the process has been established) for the contacted system, and then sends out a copy of its "state vector" to other systems (steps **106** and **107** in **Figure 8** of Jeffords). The particular system then contacts other systems and steps **106** and **107** are repeated until all systems have been attempted to be contacted.

Jeffords defines a "state vector" in col. 12, lines 52-58 as follows:

A state vector is a one-dimensional associative array of logical object name to logical object state. Each name is an "index" into the vector, and each state is stored in a "slot" associated with an index. A state vector is generated by an object and describes what that object thinks the state of all objects in the vector is. It is the object's view of itself and "relative" view of all other objects.

Jeffords does not disclose, in col. 14, line 36-col. 15, line 17 or elsewhere in the patent, "an object pool server connected to said client through said communication network and connected to said application server for pooling a proxy object corresponding to said actual object and for holding actual object management information that is information on said actual object, wherein said application server notifies said object pool server of an event according to a change in status of said application", and that the object pool server "automatically updates said actual object management information according to the notification of said event from said application server". Jeffords, instead, appears to be concerned with contacting other systems in a network, and then updating its own state vector and sending a copy of the state vector to the other contacted systems.

In responding to Applicants' arguments presented in the Response to Office Action dated June 28, 2004, the Examiner states, in part:

> Jeffords describes stored object information from the resources at col. 5, lines 37-49. The stored resource objects states are continually stored in the distributed memory and when the Replicated Resource Management objects are initialized they communicate with the resource objects.

Final Office Action dated November 3, 2004, page 5.

Col. 5, lines 37-49 of Jeffords is as follows:

> Once the RRM subsystem is initialized with the above information, its services may be used by the application. At this point, the application may do the following:
>
> examine/use the resource objects in the resource pools
> instantiate new resource objects in the resource pools
> delete resource objects from the resource pools
> change the attributes of resource objects
> receive attribute change notifications for resource objects
> use object-level messaging and object-level remote procedure calls
> receive contact lost/established notifications

This recitation states only that following initialization of the RRM system, an application may perform various actions with respect to resource objects. The recitation is not a teaching of an "application server notifies said object pool server of an event according to a change in status of said application, and said object pool server automatically updates said actual object management information according to the notification of said event from said application server" as recited in claim 1. Only the present application contains such a disclosure.

For at least all the above reasons, Jeffords does not anticipate claim 1, and claim 1 should be allowable over Jeffords in its present form.

Claim 2 depends from and further restricts claim 1, and is also not anticipated by Jeffords, at least by virtue of its dependency.

Independent claim 3 is as follows:

3. An object pool using distributed objects, comprising:
a client request analyzing unit for analyzing an access request to an object;
an object information storage unit for storing object information at the termination process of said object pool;
an object creating unit for creating an object at the starting process of said object pool according to said object information stored by said object information storage unit; and
an object managing unit for pooling the object created by said object creating unit before accessing said object from said client.

Jeffords does not disclose "an object information storage unit for storing object information at the termination process of said object pool", and does not disclose "an object creating unit for creating an object at the starting process of said object pool according to said object information stored by said object information storage unit", and "an object managing unit for pooling the object created by said object creating unit before accessing said object from said client" as recited in claim 1.

In Jeffords, a method is disclosed that is primarily involved in establishing contact with other systems. Jeffords does not disclose storing object information at a termination process of an object pool, creating an object at a starting process of an object pool, and pooling the created object before accessing the object from a client.

Claim 3, accordingly, is also not anticipated by Jeffords, and withdrawal of the rejection of claim 3 as being anticipated by Jeffords is also respectfully requested.

Claim 4 depends from and further restricts claim 3 and is also not anticipated by Jeffords, at least by virtue of its dependency.

Independent claims 5, 10, 11, 12 and 14 recite limitations similar to claim 1, and are not anticipated by Jeffords for similar reasons as discussed above with respect to claim1.

Claims 6 and 7 depend from claim 5 and claim 13 depends from claim 12, and these claims are also not anticipated by Jeffords, at least by virtue of their dependency.

Independent claim 8 contains limitations similar to claim 3, and is not anticipated by Jeffords for similar reasons as discussed above with respect to claim 3.

Claim 9 depends from and further restricts claim 8, and is also not anticipated by Jeffords, at least by virtue of its dependency.

Independent claim 15 reads as follows:

> 15. A program sending apparatus, comprising:
> a storage unit for storing a program which makes a computer execute an object pooling process for pooling, on a server, objects associated with execution of an application utilizing distributed objects, an information storing process for storing object information in said server, and a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process; and
> a sending unit for reading out said program from said storage unit, and sending said program.

Jeffords does not disclose "a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process". Jeffords nowhere discusses determining a sequence of objects to be created and nowhere discloses that such a sequence is according to object information stored by an information storing process.

In responding to Applicants' arguments regarding claim 15 in the Response to Office Action dated June 28, 2004, the Examiner states, in part:

> Jeffords states that when initialized the RRM object examines/uses the resource objects in the resource pools on col. 5, lines 37-4. This plurality of objects can be considered a sequence. These resource objects are stored in the distributed memory space by some form of information storing process.

Final Office Action dated November 3, 2004, page 5.

Applicants respectfully disagree. Applicants submit that because the RRM is able to examine/use resource objects in a resource pool, it does not suggest "a creation sequence determining process for determining a sequence of objects to be created according to said object information stored by said information storing process" as recited in claim 15.

Claim 15, accordingly, is also not anticipated by Jeffords and withdrawal of the rejection thereunder is respectfully requested.

Therefore, the rejection of claims 1-15 under 35 U.S.C. § 102 has been overcome.

Furthermore, Jeffords does not teach, suggest, or give any incentive to make the needed changes to reach the presently claimed invention. Accordingly, one of ordinary skill in the art would not be led to modify Jeffords to reach the present invention when the reference is examined as a whole. Absent some teaching, suggestion, or incentive to modify Jeffords in this manner, the presently claimed invention can be reached only through an improper use of hindsight using the Applicants' disclosure as a template to make the necessary changes to reach the claimed invention.

II. Conclusion

For all the above reasons, Applicants believe that claims 1-15 are patentable over Jeffords, and that this application is now in condition for allowance. It is, accordingly, respectfully requested that the Examiner so find and issue a Notice of Allowance in due course.

The Examiner is invited to call the undersigned at the below-listed telephone number if in the opinion of the Examiner such a telephone conference would expedite or aid the prosecution and examination of this application.

DATE: December 29, 2004

Respectfully submitted,

[illegible]

Gerald H. Glanzman
Reg. No. 25,035
Yee & Associates, P.C.
P.O. Box 802333
Dallas, TX 75380
(972) 385-8777
Attorney for Applicants